תבניות באסמבלי – ליחידה חמישית

מאת

ישראל אברמוביץ

כללי

מטרת התרגילים היא יצירת תבניות לשימוש התלמידים.

התלמידים מגיעים ליחידה החמישית לאחר לימוד יסודות מדעי המחשב, והם רגילים לחשוב על אלגוריתמים בשפה עילית, לכן יש כאן פתרונות בשפה עילית ויישומם באסמבלי.

בדף העבודה יש תירגול בסביבת העבודה לשפת C# אך היא מתאימה גם לשפת Java וגם לשפות עיליות אחרות.

הוראות קפיצה מותנית

if if-else

באסמבלי הלוגיקה היא הפוכה משפה עילית, כלומר בשפה עילית אם בהוראת הביצוע מותנה (if) התנאי מתקיים ("אמת" true) נכנסים לקטע הקוד בסוגריים המסולסלים, ואילו באסמבלי הלוגיקה היא אם התנאי **לא** מתקיים קפוץ לתגית, אחרת המשך לשורה הבאה

חישוב ערך מוחלט

ASSEMBLY	C#

JMP START
X DB ?
START:
 	
CMP X,0
JS NEG_NUM
JMP SOF
 	
NEG_NUM: NEG X
 	
SOF:
	int x = int.Parse(Console.ReadLine());
	
	bool b = x < 0;
	if(b)
	{
	x = -x;
	}

דרך נוספת

ASSEMBLY	C#

JMP START
 	
X DB ?
START:
CMP X,0
JNS SOF ; if X not Negative jump to SOF
 	
NEG X
 	
SOF:
	int x = int.Parse(Console.ReadLine());
	
	bool b = x >= 0;
	if(!b)
	{
	x = -x;
	}

במקום שימוש בפקודה JS (קפוץ אם המספר שלילי) אפשר להשתמש בהוראות הקפיצה המותנית למספרים מכוונים: JG JL JGE JLE

או בהוראות הקפיצה המותנית למספרים בלתי מכוונים: JA JB JAE JBE

if-else

השוואת ערכים של 2 משתנים והדפסת הודעה אם המספרים שווים או שונים

ASSEMBLY	C#

STR1 DB 'Equal$'
STR2 DB 'Not Equal$'
START:
 	
CMP AX,BX
JE EQUAL
; Not Equal
MOV DX, OFFSET STR2
MOV AH,9
INT 21H
JMP CONT
 	
EQUAL:
; Equal
MOV DX, OFFSET STR1
MOV AH,9
INT 21H
 	
CONT:
	int AX = 3, BX = 4;
	if(AX==BX)
	{
	  Console.WriteLine("Equal");
	}
	else
	{
	  Console.WriteLine("Not Equal");
	}

ביצוע חוזר (לולאות)

מימוש לולאה בעזרת מונה לולאה

i / AL – משמש כמונה הלולאה שרצה count פעמים.

ASSEMBLY	C#

COUNT DB 3
 	
MOV AL,0
 	
AGAIN:
 	
CMP AL,COUNT
JAE SOF
 	
; גוף הלולאה
 	
INC AL
JMP AGAIN
 	
SOF:
	int count = 3;
	int i = 0;
	
	for (; i < count;)
	{
	i++;
	;// גוף הלולאה
	}

מימוש לולאת while

לולאת while רצה כל עוד מתקיים תנאי מסויים, לדוגמה הלולאה מחברת את מספרי קלט כל עוד הסכום (הצובר) קטן מ 10.

ASSEMBLY	C#

SUM DB 0
 	
AGAIN:
 	
CMP SUM,10
JAE SOF
 	
; גוף הלולאה
 	
MOV AH,1
INT 21H
SUB AL,'0'
ADD SUM,AL
 	
JMP AGAIN
 	
SOF:
	int sum = 0;
	while(sum < 10)
	{
	int num = int.Parse(Console.ReadLine());
	sum = sum + num;
	}

מימוש לולאת do-while

גם לולאת do-while רצה כל עוד מתקיים תנאי מסויים,
אך בלולאת do-while התנאי נבדק לאחר ביצוע גוף הלולאה, כלומר גוף הלולאה ירוץ לפחות פעם אחת.
לדוגמה הלולאה מחברת את מספרי קלט כל עוד הסכום (הצובר) קטן מ 10.

ASSEMBLY	C#

SUM DB 0
 	
AGAIN:
 	
;
 	
MOV AH,1
INT 21H
SUB AL,'0'
ADD SUM,AL
 	
CMP SUM,10
JB AGAIN
 	
SOF:
	int sum = 0;
	
	do
	{
	int num = int.Parse(Console.ReadLine());
	sum = sum + num;
	}
	while (sum < 10);

מימוש לולאת for בעזרת פקודת LOOP

בלולאת LOOP מונה הלולאה הוא אוגר CX

ASSEMBLY	C#

MOV CX,10
AGAIN:
 	
;
LOOP AGAIN
	for (int i = 0; i < 10; i++)
	{
	;
	}

לולאה מקוננת

מכיוון שבלולאת LOOP מונה הלולאה הוא אוגר CX, יש בעיתיות בשימוש for בתוך for
ניתן להתגבר על זה בכמה דרכים:

א. שמירה במחסנית או במשתנה אחר את הערך של מונה הלולאה החיצונית.

ASSEMBLY	C#

MOV CX,4
AGAIN1:
PUSH CX
 	
MOV CX,3
AGAIN2:
 	
; גוף הלולאה
LOOP AGAIN2
 	
POP CX
LOOP AGAIN1
	for (int i = 0; i < 4; i++)
	{
	for (int j = 0; j < 3; j++)
	{
	; // גוף הלולאה
	}
	}

ב. מימוש של לפחות לולאה אחת, באחת הדרכים האחרות שפורטו

ASSEMBLY	C#

MOV CX,4
AGAIN1:
 	
MOV AX,3
AGAIN2:
 	
;
 	
DEC AX
CMP AX,0
JA AGAIN2
 	
LOOP AGAIN1
	for (int i = 0; i < 4; i++)
	{
	for (int j = 0; j < 3; j++)
	{
	; // גוף הלולאה
	}
	}

מערך – קטע בזכרון

- בשפת אסמבלי ההתיחסות אל מערך הינה כאל רצף של תאים (משתנים) בזכרון
- הקצאת מקום בזכרון (למשל למערך), לא מאפסת את הערכים באותו איזור זכרון (אלא אם כן זה הוגדר מראש)

איפוס מערך

בדוגמה זו איברי המערך הם מסוג בית BYTE

ASSEMBLY	C#

ARR DB 5 DUP(?)
 	
START:
 	
MOV CX,5
MOV SI,0
 	
AGAIN:
MOV ARR[SI],0
INC SI
 	
LOOP AGAIN
	int[] arr = new int[5];
	
	for (int i = 0; i < arr.Length; i++)
	{
	arr[i] = 0;
	}

ביצוע ערך מוחלט על איברי המערך

בדוגמה זו ההתיחסות את המערך הינה כאל רצף של תאים בזכרון, ואיברי המערך הם מסוג מילה WORD

ASSEMBLY	C#

ARR DW 7, -8, 11, 0, -2
 	
START:
 	
MOV CX,5
LEA SI,ARR
 	
AGAIN:
CMP [SI],0
JGE NOT_NEG
NEG WORD PTR[SI]
 	
NOT_NEG:
ADD SI,2
 	
LOOP AGAIN
	int[] arr = {7, -8, 11, 0, -2 };
	
	for (int i = 0; i < arr.Length; i++)
	{
	if(arr[i] < 0)
	arr[i] = Math.Abs(arr[i]);
	}

פרוצדורה + קריאה לפרוצדורה

פרוצדורה שלא מקבלת פרמטרים ולא מחזירה ערך

הפרוצדורה קולטת תו מהמשתמש ומדפיסה את התו העוקב

ASSEMBLY	C#

CALL PrintKelet
ret
 	
PROC PrintKelet
 	
MOV AH,1
INT 21H
 	
MOV DL,AL
INC DL
MOV AH,2
INT 21H
ret
 	
ENDP PrintKelet
	static void PrintKelet()
	{
	int x = int.Parse(Console.ReadLine());
	Console.WriteLine(x);
	}
	
	static void Main(string[] args)
	{
	PrintKelet();
	}

פרוצדורה שמקבלת פרמטר על ידי ערך (by value) ולא מחזירה ערך

הפרוצדורה מקבלת (באמצעות המחסנית) ערך של תו (ASCII) ומדפיסה אותו

ASSEMBLY	C#

MOV AL,'D'
MOV AH,0
PUSH AX
 	
CALL Print
ret
 	
PROC Print
 	
  MOV BP,SP
  MOV DX,[BP+2] ; DL contains the ASCII char
 	
  MOV AH,2
  INT 21H
  ret 2
 	
ENDP Print
	static void Print(char ch)
	{
	  Console.WriteLine(ch);
	}
	
	static void Main(string[] args)
	{
	  Print('D');
	}

פרוצדורה שמקבלת פרמטרים על ידי כתובת (by reference) ולא מחזירה ערך

הפרוצדורה מקבלת (באמצעות המחסנית) הפניה למערך (את הכתובת של התא הראשון במערך) והפניה למשתנה SUM ומעדכנת במשתנה SUM את סכום האיברים במערך

ASSEMBLY	C#

JMP START
ARR DW 7,11,8
SUM DW 0
START:
 	
LEA BX,ARR
PUSH BX
LEA BX,SUM
PUSH BX
CALL ArrSum
ret
 	
PROC ArrSum
 	
MOV BP,SP
MOV BX,[BP+2] ; BX contains the address of SUM
MOV SI,[BP+4] ; SI contains the address of ARR
 	
MOV AX,[SI]
ADD AX,[SI+2]
ADD AX,[SI+4]
 	
MOV [BX],AX
ret 4
 	
ENDP ArrSum
	static void ArrSum(int[] arr, ref int sum)
	{
	for (int i = 0; i < arr.Length; i++)
	{
	sum += arr[i];
	}
	}
	
	static void Main(string[] args)
	{
	int[] arr = {7,11,8 };
	int sum = 0;
	ArrSum(arr,ref sum);
	Console.WriteLine(sum);
	}

פעולה שמקבלת פרמטרים ומחזירה ערך

פעולה הינה פרוצדורה שמחזירה ערך בד"כ על ידי האוגרים: AL, AX, DX:AX להחזרת בית, מילה או מילה כפולה בהתאמה

ASSEMBLY	C#

JMP START
  X DW 3
  Y DW 4
START:
 	
  PUSH X
  PUSH Y
  CALL CalcRecArea
  ret
 	
PROC CalcRecArea
 	
  MOV BP,SP
  MOV AX,[BP+2] ;
  MOV BX,[BP+4] ;
  MUL BL ; AX = AL*BL
 	
  ret 4
 	
ENDP CalcRecArea
	static int CalcRecArea(int x, int y)
	{
	  return x * y;
	}
	
	static void Main(string[] args)
	{
	  int x = 3, y = 4;
	  int AX = CalcRecArea(x,y);
	  Console.WriteLine(AX);
	}